P- 23

NAG 2-827

CENTER FOR SPACE STRUCTURES AND CONTROLS

PROGRESS REPORT: PART II
A Transient FETI Methodology For Large-Scale Parallel Implicit Computations in Structural Mechanics

by

CHARBEL FARHAT

AUGUST 1993

COLLEGE OF ENGINEERING
UNIVERSITY OF COLORADO
CAMPUS BOX 429
BOULDER, COLORADO 80309

A TRANSIENT FETI METHODOLOGY FOR LARGE-SCALE PARALLEL IMPLICIT COMPUTATIONS IN STRUCTURAL MECHANICS

Charbel FARHAT and Luis CRIVELLI
Department of Aerospace Engineering Sciences
and Center for Space Structures and Controls
University of Colorado at Boulder
Boulder, CO 80309-0429, U. S. A.

Explicit codes are often used to simulate the nonlinear dynamics of large-scale structural systems, even for low frequency response, because the storage and CPU requirements entailed by the repeated factorizations traditionally found in implicit codes rapidly overwhelm the available computing resources. With the advent of parallel processing, this trend is accelerating because explicit schemes are also easier to parallelize than implicit ones. However, the time step restriction imposed by the Courant stability condition on all explicit schemes cannot yet — and perhaps will never — be offset by the speed of parallel hardware. Therefore, it is essential to develop efficient and robust alternatives to direct methods that are also amenable to massively parallel processing because implicit codes using unconditionally stable time-integration algorithms are computationally more efficient than explicit codes when simulating low-frequency dynamics. Here we present a domain decomposition method for implicit schemes that requires significantly less storage than factorization algorithms, that is several times faster than other popular direct and iterative methods, that can be easily implemented on both shared and local memory parallel processors, and that is both computationally and communication-wise efficient. The proposed transient domain decomposition method is an extension of the method of Finite Element Tearing and Interconnecting (FETI) developed by Farhat and Roux for the solution of static problems. Serial and parallel performance results on the CRAY Y-MP/8 and the iPSC-860/128 systems are reported and analyzed for realistic structural dynamics problems. These results establish the superiority of the FETI method over both the serial/parallel conjugate gradient algorithm with diagonal scaling and the serial/parallel direct method, and contrast the computational power of the iPSC-860/128 parallel processor with that of the CRAY Y-MP/8 system.

[4], Belytschko, Plaskacz, Kennedy and Greenwell [5], and Biffle [6]) because these methodologies (a) are easier to parallelize than implicit schemes and direct solvers, and (b) they usually induce short range interprocessor communications that are relatively inexpensive, while the factorization methods used in most implicit schemes induce long range interprocessor communications that often ruin the sought-after speed-up. However, the time step restriction imposed by the Courant stability condition on all explicit schemes cannot yet — and perhaps will never — be offset by the speed of parallel hardware, and many iterative solvers often fail when applied to systems arising from the analysis of large-scale flexible structures. Therefore, it is essential to develop efficient and robust alternatives to direct methods that are also amenable to massively parallel processing, because unconditionally stable implicit schemes are computationally more efficient than explicit time-integration methodologies at simulating low-frequency dynamics. The EBE/PCG algorithm developed by Hughes and his co-workers [1, 2] is such an alternative which additionally utilizes the structure inherent in finite element formulations and implementations. Here, we propose another alternative which is driven by substructuring concepts and which achieves a greater improvement over the CG algorithm with diagonal scaling than reported in [1] for the EBE/PCG scheme. However, unlike EBE methods, the proposed methodology requires the assembly and factorization of substructure level matrices and therefore requires more storage than the EBE/PCG solution algorithm.

A good balance between direct and iterative solution algorithms is provided by Domain Decomposition (DD) methods which usually blend both of these solution strategies. A well designed DD method requires less storage than direct solvers and converges faster than other purely iterative algorithms when applied to highly ill-conditioned systems (see, for example, the collection of papers compiled in [7]). Moreover, in their simplest form, DD methods have an explicit character because they effectively share information only between neighboring subdomains, which makes them very attractive for parallel processing. The method of Finite Element Tearing and Interconnecting (FETI) is a DD algorithm based on a hybrid variational principle that was developed by Farhat and Roux [8] for the parallel solution of self-adjoint elliptic partial differential equations. It combines a direct solver for computing the incomplete subdomain displacement fields with a PCG algorithm for extracting the dual tractions at the subdomain interfaces. This method was shown to outperform optimized direct solvers on both serial and coarse-grained multiprocessors such as the CRAY Y-MP/8 system, and to compare favorably with other DD algorithms on a 32 processor iPSC-2 (Farhat and Roux [9]). In this paper, we present an extension of the FETI methodology to structural dynamics problems. In particular, we construct two subdomain-by-

stress and strain tensors, ρ^s is the mass density, u is the displacement field, f is the forcing field, and $\lambda^{s,q}$ is a Lagrange multiplier function which represents the interface tractions that maintain equilibrium between Ω^s and a neighboring Ω^q (interconnecting). The first of the inter-substructure continuity constraints (Eqs. (4)) can be dualized as:

$$\int_{\overline{\Omega}^s \bigcap \overline{\Omega}^q} \delta\lambda^{s,q}(u^s - u^q)\, d\Gamma \;=\; 0 \tag{5}$$

If the original mesh of the global structure does not contain incompatible elements, the subdomains $\{\Omega^s\}_{s=1}^{s=N_s}$ are guaranteed to have compatible interfaces since these subdomains are obtained by partitioning the global mesh into N_s submeshes. Therefore, we assume in the sequel that discrete Lagrange multipliers are introduced at the subdomain interfaces to enforce the inter-substructure displacement continuity. The piece-wise continuous approximation of the Lagrange multipliers $\lambda^{s,q}$ in Eq. (5) is treated in Farhat and Geradin [10] for static problems. Using a standard Galerkin procedure where the displacement field is approximated by suitable shape functions as:

$$u^s \;=\; \mathbf{N}\mathbf{u}^s \tag{6}$$

and linearizing the equations of dynamic equilibrium around $\mathbf{u}_{n+1}$, Eqs. (3) and (5) are transformed into the following algebraic system:

$$\begin{aligned} \mathbf{M}^s \Delta\ddot{\mathbf{u}}_{n+1}^{s(k+1)} + \mathbf{K}^{t^s} \Delta\mathbf{u}_{n+1}^{s(k+1)} \;&=\; \mathbf{r}_{n+1}^{s(k)} - \mathbf{B}^{s^T} \boldsymbol{\lambda}_{n+1}^{(k+1)} \qquad\qquad s = 1,\ ...,\ N_s \\ \sum_{s=1}^{s=N_s} \mathbf{B}^s \Delta\mathbf{u}_{n+1}^{s(k+1)} \;&=\; 0 \end{aligned} \tag{7}$$

where the superscript ^{T} indicates a transpose, $\mathbf{M}^s$ and $\mathbf{K}^{t^s}$ are respectively the subdomain mass and tangent stiffness matrices, $\Delta\mathbf{u}_{n+1}^{s(k+1)}$ and $\mathbf{r}_{n+1}^{s(k)}$ are respectively the subdomain vector of nodal displacement increments and the subdomain vector of out-of-balance nodal forces, $\mathbf{B}^s$ is a boolean matrix that describes how $\overline{\Omega}^s$ is connected to the global interface, and $\boldsymbol{\lambda}_{n+1}^{(k+1)}$ is the vector of Lagrange multiplier unknowns at iteration $k+1$ and step $n+1$. For linear elastostatic problems, Eqs. (7) above correspond to the discretization of a saddle-point variational principle whose mathematical and computational properties are analyzed in Farhat and Roux [8, 9].

displacement field increment $\Delta \mathbf{u}_{n+1}^{(k+1)}$ and the momentum increment $\Delta \mathbf{v}_{n+1}^{(k+1)} = \overline{\mathbf{M}}\, \Delta \dot{\mathbf{u}}_{n+1}^{(k+1)}$.

Here, we select a midpoint rule version of the time-integration algorithm **(A3)** and summarize it in order to keep this paper self-contained. Let $\Delta \mathbf{v}_{n+\frac{1}{2}}^{(k+1)}$ denote the momentum increment at iteration $k+1$ and at the midpoint between steps n and $n+1$:

$$\Delta \mathbf{v}_{n+\frac{1}{2}}^{(k+1)} = \overline{\mathbf{M}}\, \Delta \dot{\mathbf{u}}_{n+\frac{1}{2}}^{(k+1)} \tag{9}$$

The midpoint subdomain increments $\Delta \mathbf{u}_{n+\frac{1}{2}}^{s(k+1)}$ and $\Delta \mathbf{v}_{n+\frac{1}{2}}^{s(k+1)}$ are integrated as follows:

$$\begin{aligned} \Delta \mathbf{u}_{n+\frac{1}{2}}^{s(k+1)} &= \frac{\Delta t}{2} \Delta \dot{\mathbf{u}}_{n+\frac{1}{2}}^{s(k+1)} \\ \Delta \mathbf{v}_{n+\frac{1}{2}}^{s(k+1)} &= \frac{\Delta t}{2} \Delta \dot{\mathbf{v}}_{n+\frac{1}{2}}^{s(k+1)} \end{aligned} \tag{10}$$

where Δt is the time step. Substituting Eqs. (10) into the dynamic equations of subdomain equilibrium (7) written at the midpoint step $n+\frac{1}{2}$ leads to:

$$\Delta \mathbf{v}_{n+\frac{1}{2}}^{s(k+1)} = \frac{2}{\Delta t} \mathbf{M}^s \Delta \mathbf{u}_{n+\frac{1}{2}}^{s(k+1)} \tag{11}$$

and

$$\begin{aligned} (\mathbf{M}^s + \frac{\Delta t^2}{4} \mathbf{K}^{t^s}) \Delta \mathbf{u}_{n+\frac{1}{2}}^{s(k+1)} &= \frac{\Delta t^2}{4} (\mathbf{r}_{n+\frac{1}{2}}^{s(k)} - \mathbf{B}^{s^T} \boldsymbol{\lambda}^{(k+1)}) \qquad s = 1, ..., N_s \\ \sum_{s=1}^{s=N_s} \mathbf{B}^s \Delta \mathbf{u}_{n+\frac{1}{2}}^{s(k+1)} &= 0 \\ \mathbf{u}_{n+\frac{1}{2}}^{s(k+1)} &= \mathbf{u}_{n+\frac{1}{2}}^{s(k)} + \Delta \mathbf{u}_{n+\frac{1}{2}}^{s(k+1)} \end{aligned} \tag{12}$$

After Eqs. (12) are repeatedly solved for all nonlinear increments, $\mathbf{u}_{n+\frac{1}{2}}^{s}$ is found and the time derivative of each subdomain momentum could be obtained via back-substitution as:

$$\dot{\mathbf{v}}_{n+\frac{1}{2}}^{s} = \mathbf{f}_{n+\frac{1}{2}}^{s^{ext}} - \mathbf{B}^{s^T} \boldsymbol{\lambda} - \mathbf{f}^{s^{int}}(\mathbf{u}_{n+\frac{1}{2}}^{s}, \mathbf{p}_c, \theta) \tag{13}$$

However, in order to bypass the dynamics of the Lagrange multipliers we implicitly invoke the equilibrium condition $\sum_{s=1}^{s=N_s} \mathbf{B}^{s^T} \boldsymbol{\lambda}^{(k+1)} = 0$ and directly compute the momentum increment in assembled form as:

$$\dot{\mathbf{v}}_{n+\frac{1}{2}} = \mathbf{f}_{n+\frac{1}{2}}^{ext} - \mathbf{f}^{int}(\mathbf{u}_{n+\frac{1}{2}}, \mathbf{p}_c, \theta) \tag{14}$$

subdomain displacement fields [13]. Because of this compactness, the eigenvalues of the matrix $\sum_{s=1}^{s=N_s} \mathbf{B}^s \mathbf{K}^{t^s -1} \mathbf{B}^{s^T}$ are well separated and have a higher density towards the low end of their spectrum. For a fixed Δt, we can show that $\sum_{s=1}^{s=N_s} \mathbf{B}^s (\mathbf{M}^s + \frac{\Delta t^2}{4} \mathbf{K}^{t^s})^{-1} \mathbf{B}^{s^T}$ has similar eigen properties. The objectives of this section are: (a) to numerically illustrate the spectral properties of the transient interface operator $\sum_{s=1}^{s=N_s} \mathbf{B}^s (\mathbf{M}^s + \frac{\Delta t^2}{4} \mathbf{K}^{t^s})^{-1} \mathbf{B}^{s^T}$, and (b) to highlight the impact of these spectral properties on the convergence rate of the CG algorithm.

Consider the three-dimensional two-subdomain cantilever problem depicted in FIG. 1. The beam has a square cross section and a 10/1 length/height aspect ratio. Two finite element meshes are constructed using 8-node brick elements. The first mesh, M_1, contains 2400 internal degrees of freedom (d.o.f.) and 192 interface d.o.f. The second mesh, M_2, is finer: it contains 16320 internal d.o.f. and 672 interface d.o.f.

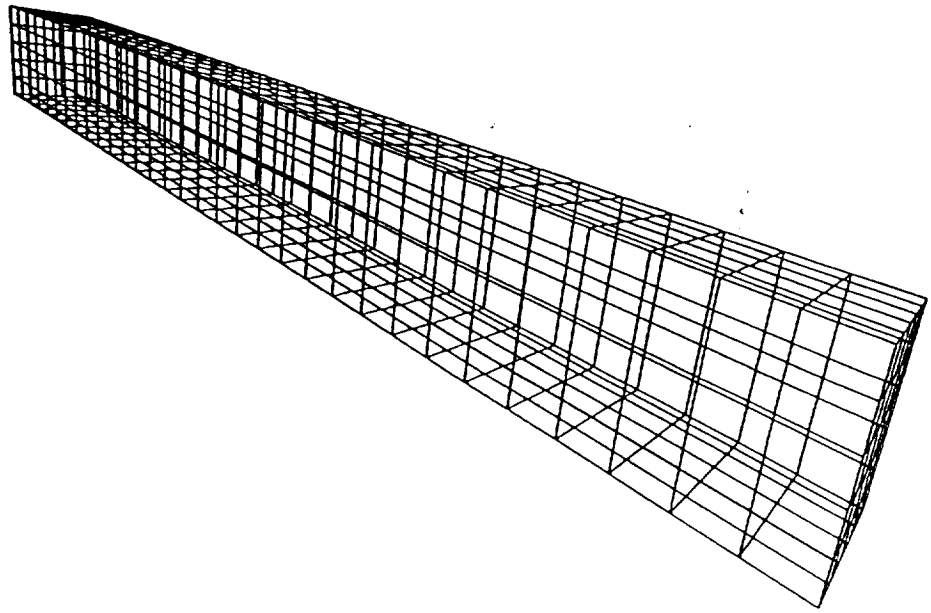

FIG. 1 *A three-dimensional cantilever problem*

The x axis of the bar diagrams depicted in FIG. 2-3 represents the eigenvalues scaled by the smallest eigenvalue, and the y axis the number of eigenvalues per interval. For both meshes, $\widetilde{\mathbf{F}}_I^t$ is shown to have a few large eigenvalues that are well separated from the small ones. Figure 3 suggests that this separation amplifies when the mesh size $h \to 0$. Indeed, one can mathematically prove that the large eigenvalues of $\widetilde{\mathbf{F}}_I^t$ stabilize when the mesh size decreases, while its small eigenvalues accumulate towards zero. Essentially, this is because $\widetilde{\mathbf{F}}_I^t$ involves the inverses of the pencils $(\mathbf{M}^s, \mathbf{K}^{t^s})$ and therefore its high modes correspond to physical modes while its low modes correspond to mesh modes.

The spectral distribution of the interface problem associated with the FETI methodology has important consequences on the convergence rate of the CG algorithm. During the first iterations, the conjugate gradient algorithm mostly captures the eigenvectors associated with the large eigenvalues. Since the high numerical modes of $\widetilde{\mathbf{F}}_I^t$ correspond to the low physical modes of the structure and since $\widetilde{\mathbf{F}}_I^t$ has only a few relatively high eigenvalues, the CG algorithm applied to the solution of Eq. (16) quickly gives a good approximation of the displacement increments. Intuitively, one can imagine that after the few relatively high modes of the interface operator are captured, the "effective" condition number of $\widetilde{\mathbf{F}}_I^t$ — that is the ratio of its largest uncaptured eigenvalue over its smallest one — becomes significantly smaller than the original condition number of $\widetilde{\mathbf{F}}_I^t$, which accelerates the convergence of the CG algorithm. A detailed analysis of this superconvergence behavior of the CG algorithm in the presence of well separated eigenvalues can be found in the work of van der Sluis and van der Vorst [14].

The impact of the spectral distribution of the transient interface operator $\widetilde{\mathbf{F}}_I^t$ on the convergence rate of the CG algorithm is highlighted in TABLE 1 which reports the number of iterations to achieve convergence for the FETI methodology described in this paper and for the classical Schur complement method. The transient interface operator resulting from the latter DD method (static condensation) is denoted here by $\widetilde{\mathbf{S}}_I^t$. While both $\widetilde{\mathbf{F}}_I^t$ and $\widetilde{\mathbf{S}}_I^t$ are shown to have identical two-norm condition numbers ($\kappa_2(\widetilde{\mathbf{F}}_I^t) = \kappa_2(\widetilde{\mathbf{S}}_I^t)$), the CG algorithm applied to $\widetilde{\mathbf{F}}_I^t$ is shown to converge twice as fast as when applied to $\widetilde{\mathbf{S}}_I^t$. This clearly demonstrates that conditioning is not always the only factor governing the convergence rate of the CG algorithm.

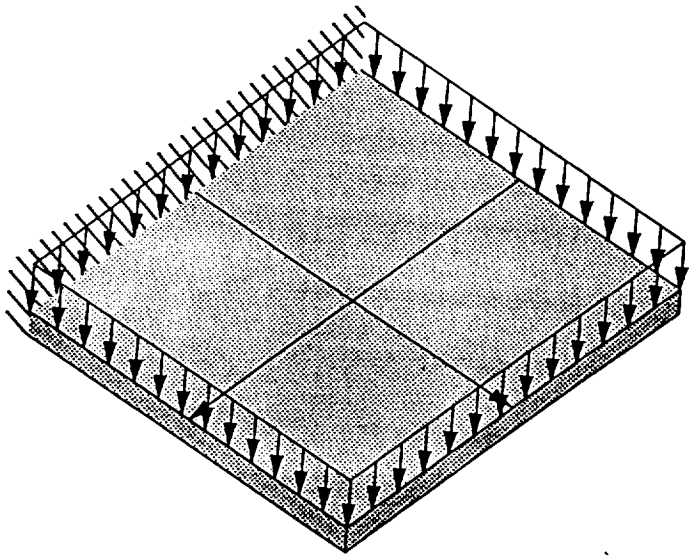

FIG. 4 *A four-subdomain bending plate problem*

TABLE 2

Four-subdomain bending plate problem

Discretization: $N \times N$ where $N = \frac{1}{h}$

Algorithm: CG-FETI

Convergence criterion: $\frac{||\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)})\ \Delta\mathbf{u}_{n+1}^{(k+1)} - \mathbf{r}(\mathbf{u}_{n+1}^{(k)})||_2}{||\mathbf{r}(\mathbf{u}_{n+1}^{(k)})||_2} \leq 10^{-4}$

h	# of d.o.f.	# of interface d.o.f.	# of iterations
$\frac{1}{10}$	605	55	43
$\frac{1}{20}$	2205	105	54
$\frac{1}{40}$	8405	205	74
$\frac{1}{80}$	32805	405	75
$\frac{1}{160}$	129605	805	75

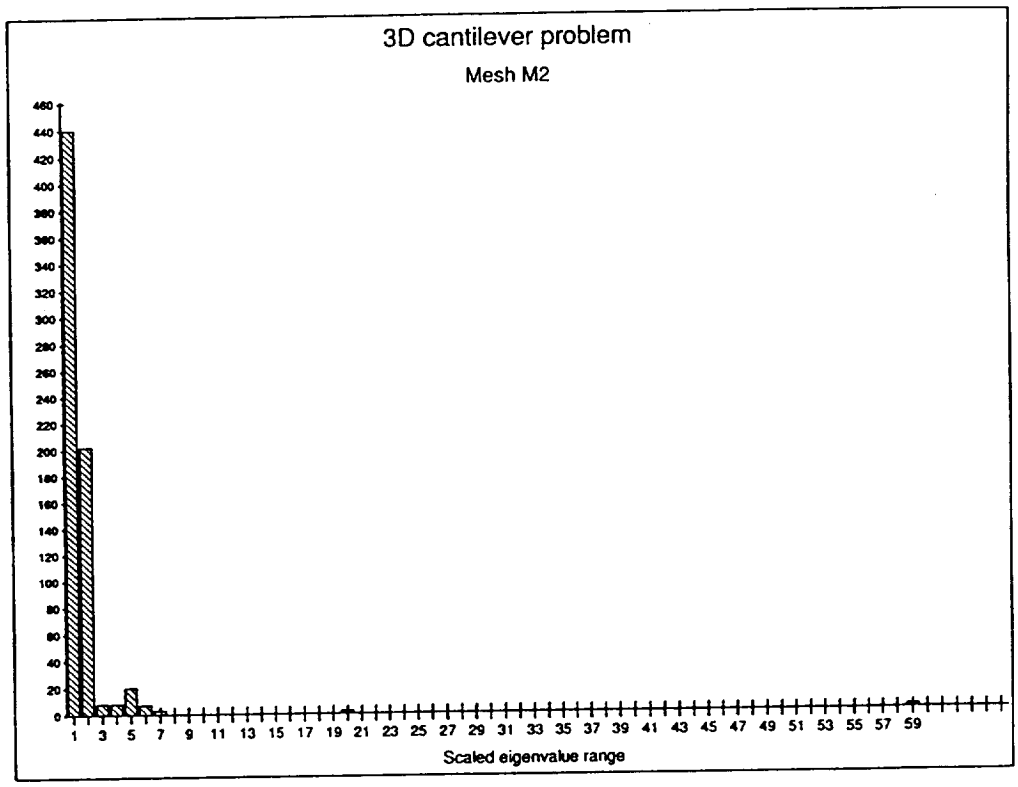

FIG. 5 *Spectral density of* $\widetilde{\mathbf{L}}_I^{t^{-1}}\widetilde{\mathbf{F}}_I^t$ *for mesh* M_2

By comparing FIG. 3 and FIG. 5, the reader can observe that $\mathbf{L}_I^{t^{-1}}$ essentially amplifies the separation between the clusters of small and large eigenvalues of the interface problem, which accelerates the convergence of the CG algorithm. Following the reasoning outlined in Section 3, the reader can also conclude from FIG. 3 and FIG. 5 that after the few large eigenvalues of the interface problem are captured, the "effective" condition number of $\mathbf{L}_I^{t^{-1}}\widetilde{\mathbf{F}}_I^t$ is much smaller than that of $\widetilde{\mathbf{F}}_I^t$. This "superconvergence" behavior is highlighted in TABLE 3 below for the three-dimensional two-subdomain cantilever problem.

matrix-vector products of sizes equal to the subdomain interfaces. From a mechanical viewpoint, solving Eq. (19) corresponds to finding a set of "lumped" interface forces that can reproduce the imposed jump of the displacement increments. Therefore, the problem described by Eq. (20) is indeed an approximate inverse of the interface problem.

5. Preconditionig with a primal operator

5.1 Sum of the exact inverses

For the sake of clarity and notation simplicity, we first assume that the mesh partition does not contain cross-points — that is, points where more than two subdomains intersect. After the mechanical interpretation of the primal preconditioner presented below is given in Section 5.2, the treatment of cross-points becomes straightforward.

A better approximation of $\widetilde{\mathbf{F}}_I^{t^{-1}}$ can be obtained by approximating the inverse of the sum by the sum of the exact inverses — that is, by assuming that:

$$[\sum_{s=1}^{s=N_s} \mathbf{B}^s(\mathbf{M}^s + \frac{\Delta t^2}{4}\mathbf{K}^{t^s})^{-1}\mathbf{B}^{s^T}]^{-1} \approx \sum_{s=1}^{s=N_s} [\mathbf{B}^s(\mathbf{M}^s + \frac{\Delta t^2}{4}\mathbf{K}^{t^s})^{-1}\mathbf{B}^{s^T}]^{-1} \quad (21)$$

which leads to the following preconditioner:

$$\widetilde{\mathbf{D}}_I^{t^{-1}} = \sum_{s=1}^{s=N_s} [\mathbf{B}^s(\mathbf{M}^s + \frac{\Delta t^2}{4}\mathbf{K}^{t^s})^{-1}\mathbf{B}^{s^T}]^{-1} \quad (22)$$

If the subdomain mass and stiffness matrices and the subdomain displacement field are partitioned as:

$$\mathbf{M}^s = \begin{bmatrix} \mathbf{M}^s_{ii} & \mathbf{M}^s_{ib} \\ \mathbf{M}^{s^T}_{ib} & \mathbf{M}^s_{bb} \end{bmatrix}, \quad \mathbf{K}^s = \begin{bmatrix} \mathbf{K}^s_{ii} & \mathbf{K}^s_{ib} \\ \mathbf{K}^{s^T}_{ib} & \mathbf{K}^s_{bb} \end{bmatrix}, \text{ and } \mathbf{u}^s = \begin{bmatrix} \mathbf{u}^s_i \\ \mathbf{u}^s_b \end{bmatrix}$$

where the subscripts $_i$ and $_b$ respectively refer to the internal and interface boundary degrees of freedom, the inverse of $(\mathbf{M}^s + \frac{\Delta t^2}{4}\mathbf{K}^{t^s})$ can be written as the solution of the partitioned matrix equations:

$$\begin{bmatrix} \mathbf{M}^s_{ii} + \frac{\Delta t^2}{4}\mathbf{K}^s_{ii} & \mathbf{M}^s_{ib} + \frac{\Delta t^2}{4}\mathbf{K}^s_{ib} \\ \mathbf{M}^{s^T}_{ib} + \frac{\Delta t^2}{4}\mathbf{K}^{s^T}_{ib} & \mathbf{M}^s_{bb} + \frac{\Delta t^2}{4}\mathbf{K}^s_{bb} \end{bmatrix} \begin{bmatrix} \mathbf{A}^s_{ii} & \mathbf{A}^s_{ib} \\ \mathbf{A}^{T^s}_{ib} & \mathbf{A}^s_{bb} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{I} \end{bmatrix} \quad (23)$$

Here, $\mathbf{p}_b^q$ denotes the search direction computed during the q-th iteration, and $\bar{\mathbf{r}}_b^q$ denotes the q-th residual which represents the jump of the displacement increments across the subdomain interfaces. All computations summarized in Eqs. (28) can be performed at the subdomain level as follows:

$$\begin{aligned}
\bar{\mathbf{p}}_b^{s^q} &= \mathbf{B}^s(\mathbf{M}^s + \frac{\Delta t^2}{4}\mathbf{K}^{t^s})^{-1}\mathbf{B}^{s^T}\mathbf{p}_b^{s^q} \\
\mathbf{z}_b^{s^q} &= [(\mathbf{M}_{bb}^s + \frac{\Delta t^2}{4}\mathbf{K}_{bb}^s) \\
&\quad - (\mathbf{M}_{ib}^s + \frac{\Delta t^2}{4}\mathbf{K}_{ib}^s)^T(\mathbf{M}_{ii}^s + \frac{\Delta t^2}{4}\mathbf{K}_{ii}^s)^{-1}(\mathbf{M}_{ib}^s + \frac{\Delta t^2}{4}\mathbf{K}_{ib}^s)]\bar{\mathbf{r}}_b^{s^q}
\end{aligned} \tag{29}$$

The first of Eqs. (29) can be re-formulated as a problem with *Neumann* boundary conditions (indicated between braces {}):

$$\begin{bmatrix} \mathbf{M}_{ii}^s + \frac{\Delta t^2}{4}\mathbf{K}_{ii}^s & \mathbf{M}_{ib}^s + \frac{\Delta t^2}{4}\mathbf{K}_{ib}^s \\ \mathbf{M}_{ib}^{s^T} + \frac{\Delta t^2}{4}\mathbf{K}_{ib}^{s^T} & \mathbf{M}_{bb}^s + \frac{\Delta t^2}{4}\mathbf{K}_{bb}^s \end{bmatrix} \begin{bmatrix} \bar{\mathbf{p}}_i^{s^q} \\ \bar{\mathbf{p}}_b^{s^q} \end{bmatrix} = \begin{bmatrix} 0 \\ \{\mathbf{p}_b^{s^q}\} \end{bmatrix} \tag{30}$$

The solution of this problem $\bar{\mathbf{p}}_b^{s^q}$ represents the trace on the interface boundary of Ω^s of the displacement field resulting from prescribing the interface traction forces $\mathbf{p}_b^{s^q}$.

The second of Eqs. (29) can be re-formulated as a problem with *Dirichlet* boundary conditions (indicated between braces {}):

$$\begin{bmatrix} \mathbf{M}_{ii}^s + \frac{\Delta t^2}{4}\mathbf{K}_{ii}^s & \mathbf{M}_{ib}^s + \frac{\Delta t^2}{4}\mathbf{K}_{ib}^s \\ \mathbf{M}_{ib}^{s^T} + \frac{\Delta t^2}{4}\mathbf{K}_{ib}^{s^T} & \mathbf{M}_{bb}^s + \frac{\Delta t^2}{4}\mathbf{K}_{bb}^s \end{bmatrix} \begin{bmatrix} \bar{\mathbf{z}}_i^{s^q} \\ \{\bar{\mathbf{r}}_b^{s^q}\} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{z}_b^{s^q} \end{bmatrix} \tag{31}$$

and which can be solved in two steps:

$$\begin{aligned}
\textit{Solve} \quad & (\mathbf{M}_{ii}^s + \frac{\Delta t^2}{4}\mathbf{K}_{ii}^s)\bar{\mathbf{z}}_i^{s^q} = -(\mathbf{M}_{ib}^s + \frac{\Delta t^2}{4}\mathbf{K}_{ib}^s)\bar{\mathbf{r}}_b^{s^q} \\
\textit{Evaluate} \quad & \mathbf{z}_b^{s^q} = (\mathbf{M}_{ib}^{s^T} + \frac{\Delta t^2}{4}\mathbf{K}_{ib}^{s^T})\bar{\mathbf{z}}_i^{s^q} + (\mathbf{M}_{bb}^s + \frac{\Delta t^2}{4}\mathbf{K}_{bb}^s)\bar{\mathbf{r}}_b^{s^q}
\end{aligned} \tag{32}$$

Therefore, the preconditioning step reformulated in Eq. (31) corresponds to finding in each subdomain Ω^s the traction forces that are needed to prescribe the interface boundary displacement increment jump $\bar{\mathbf{r}}_b^{s^q}$.

The mechanical interpretation of the CG algorithm with the primal preconditioner $\widetilde{\mathbf{D}}_I^t$ is straightforward. Within each iteration, a Neumann problem is first

a polynomial function of the ratio between consecutive eigenvalues.

The loss of orthogonality of the search directions during the solution of the interface problem (16) has a disastrous consequence on the convergence rate of the PCG algorithm. To remedy this problem, we introduce a reorthogonalization procedure within the PCG algorithm which however, in order to determine the new direction vector, entails at each iteration q the following additional burden:

(a) the storage of the direction vector $\mathbf{p}^q$ and the product $\widetilde{\mathbf{F}}_I^t \mathbf{p}^q$.

(b) the evaluation of q dot products of the form $\mathbf{r}_j^T[\widetilde{\mathbf{F}}_I^t \mathbf{p}^q]$ where $[\widetilde{\mathbf{F}}_I^t \mathbf{p}^q]$ is readily available and $1 \leq j < q$, and of an $n_I \times j$ matrix-vector product where n_I is the number of interface unknowns.

Clearly, such a reorthogonalization procedure is not feasible if introduced during the solution via the PCG algorithm of a global finite element problem, as it would require unreasonable amounts of memory and CPU. However, it is quite affordable within the context of a domain decomposition algorithm as it applies only to the interface problem. In particular, the reader should note that the additional computational costs outlined in (b) are small compared to the cost of the pair of forward and backward substitutions that are required at each iteration q of the PCG algorithm in order to evaluate the product $\widetilde{\mathbf{F}}_I^t \mathbf{p}^q$.

The efficiency of the reorthogonalization procedure discussed above is highlighted in TABLE 4 for the four-subdomain bending plate problem introduced in Section 3.

TABLE 4

Four-subdomain bending plate problem

Algorithm: PCG-FETI

Preconditioner: $\widetilde{\mathbf{L}}_I^{t^{-1}}$

Convergence criterion: $\frac{\|\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)})\,\Delta\mathbf{u}_{n+1}^{(k+1)} - \mathbf{r}(\mathbf{u}_{n+1}^{(k)})\|_2}{\|\mathbf{r}(\mathbf{u}_{n+1}^{(k)})\|_2} \leq 10^{-3}$

Multiprocessor: iPSC-860 (4 processors)

h	# of iterations (without reorth.)	# of iterations (with reorth.)	CPU (without reorth.)	CPU (with reorth.)
$\frac{1}{20}$	42	22	5.5 s.	3.7 s.
$\frac{1}{40}$	66	24	33.2 s.	16.5 s.

From (16) and (35), it is clear that the gradient of the interface problem (16) is indeed the jump in the displacement increments at the subdomain interfaces. Therefore, the norm of the residual of the interface problem $|| - \mathbf{G}\boldsymbol{\lambda}^{(k+1)^q} + \mathbf{H}||$ gives the order of magnitude of the error in $\Delta \mathbf{u}_{n+\frac{1}{2}}$ and not the order of magnitude of the global residual $\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)})\ \Delta \mathbf{u}_{n+\frac{1}{2}}^{(k+1)} - \mathbf{r}(\mathbf{u}_{n+1}^{(k)})$, where $\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)}) = \mathbf{M} + \frac{\Delta t^2}{4}\mathbf{K}^t$. Moreover, since the condition number of $\widetilde{\mathbf{K}}^t$ varies as $O(1/h^2)$ while the condition number of $\widetilde{\mathbf{F}}_I^t$ varies as $O(1/h)$, the convergence criterion:

$$|| - \mathbf{G}\boldsymbol{\lambda}^{(k+1)^q} + \mathbf{H}|| \ \leq\ \epsilon\ || - \mathbf{G}\boldsymbol{\lambda}^{(k+1)^0} + \mathbf{H}|| \tag{36}$$

does not guarantee that:

$$||\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)})\ \Delta \mathbf{u}_{n+\frac{1}{2}}^{(k+1)^q} - \mathbf{r}(\mathbf{u}_{n+1}^{(k)})|| \ \leq\ \epsilon\ ||\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)})\ \Delta \mathbf{u}_{n+\frac{1}{2}}^{(k+1)^0} - \mathbf{r}(\mathbf{u}_{n+1}^{(k)})|| \tag{37}$$

We have observed that for most structural problems, the norm of the relative global residual (37) is typically 10^2 to 10^3 larger than the norm of the relative interface residual (36), which clearly indicates that the convergence of the FETI methodology should be based on the global criterion (37).

Unfortunately, evaluating (37) at every PCG iteration q requires performing a global matrix-vector multiply in order to compute the global residual $\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)})\ \Delta \mathbf{u}_{n+\frac{1}{2}}^{(k+1)^q} - \mathbf{r}(\mathbf{u}_{n+1}^{(k)})$. This would double both the computational and communication costs of a PCG iteration. Therefore, we had to develop an estimator for the global residual that is accurate and computationally economical.

Using Eq. (12) and the partitioning introduced in Section 5.1, the restriction of the global residual to every subdomain Ω^s can be written as:

$$\widetilde{\mathbf{K}}^{t^s}(\mathbf{u}_{n+1}^{s^{(k)}})\ \Delta \mathbf{u}_{n+\frac{1}{2}}^{s^{(k+1)^q}} - \mathbf{r}(\mathbf{u}_{n+1}^{s^{(k)}}) = \begin{bmatrix} (\mathbf{M}_{ib}^s + \frac{\Delta t^2}{4}\mathbf{K}_{ib}^s)\ (\overline{\Delta \mathbf{u}_{n+\frac{1}{2}\ bb}^{s^{(k+1)^q}}} - \Delta \mathbf{u}_{n+\frac{1}{2}\ bb}^{s^{(k+1)^q}}) \\ (\mathbf{M}_{bb}^s + \frac{\Delta t^2}{4}\mathbf{K}_{bb}^s)\ (\overline{\Delta \mathbf{u}_{n+\frac{1}{2}\ bb}^{s^{(k+1)^q}}} - \Delta \mathbf{u}_{n+\frac{1}{2}\ bb}^{s^{(k+1)^q}}) \end{bmatrix} \tag{38}$$

where $\overline{\Delta \mathbf{u}_{n+\frac{1}{2}\ bb}^{s^{(k+1)^q}}} - \Delta \mathbf{u}_{n+\frac{1}{2}\ bb}^{s^{(k+1)^q}}$ is the jump in the displacement increments at the subdomain boundary interface. Clearly, the reaction forces $(\mathbf{M}_{ib}^s + \frac{\Delta t^2}{4}\mathbf{K}_{ib}^s)\ (\overline{\Delta \mathbf{u}_{n+\frac{1}{2}\ bb}^{s^{(k+1)^q}}} - \Delta \mathbf{u}_{n+\frac{1}{2}\ bb}^{s^{(k+1)^q}})$ are zero except on those degrees of freedom that are connected to the interface ones. Moreover, equilibrium suggests that $||(\mathbf{M}_{ib}^s + \frac{\Delta t^2}{4}\mathbf{K}_{ib}^s)\ (\overline{\Delta \mathbf{u}_{n+\frac{1}{2}\ bb}^{s^{(k+1)^q}}} - \Delta \mathbf{u}_{n+\frac{1}{2}\ bb}^{s^{(k+1)^q}})||$ and $||(\mathbf{M}_{bb}^s + \frac{\Delta t^2}{4}\mathbf{K}_{bb}^s)\ (\overline{\Delta \mathbf{u}_{n+\frac{1}{2}\ bb}^{s^{(k+1)^q}}} - \Delta \mathbf{u}_{n+\frac{1}{2}\ bb}^{s^{(k+1)^q}})||$ are of the same order

The discrepancy between the interface and global residuals and the accuracy of the proposed global residual estimator are demonstrated in FIG. 6 which reports the logarithm of the Euclidean norm of the various residuals for the four subdomain bending plate problem with $h = 1/40$ (Section 3).

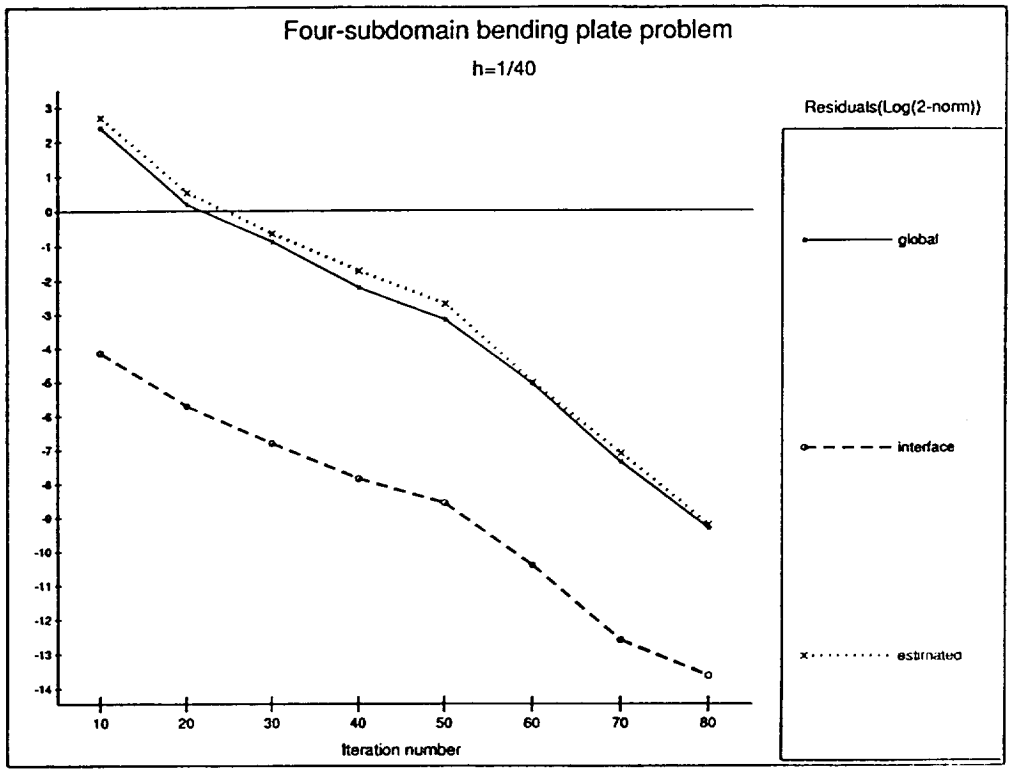

FIG. 6 *Accuracy of the global residual estimator*

7.2. Setting the tolerance for the PCG algorithm

Let $\mathbf{d}_{n+1}$, $\mathbf{u}_{n+1}^k$, and $(\mathbf{u}_{n+1}^k)^\infty$ denote respectively the exact solution of the nonlinear problem (1) at time step $n+1$, the exact solution of the linearized problem at the k-th nonlinear iteration of time step $n+1$, and the PCG solution of the linearized problem at the k-th nonlinear iteration of time step $n+1$. Our stopping criterion for the PCG algorithm is:

$$||\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)})\,\Delta\mathbf{u}_{n+\frac{1}{2}}^{(k+1)^q} - \mathbf{r}(\mathbf{u}_{n+1}^{(k)})|| \leq \epsilon\,||\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)})\,\Delta\mathbf{u}_{n+\frac{1}{2}}^{(k+1)^0} - \mathbf{r}(\mathbf{u}_{n+1}^{(k)})|| \quad (44)$$

where q denotes the PCG iteration number. It follows that:

$$||(\mathbf{u}_{n+1}^{k+1})^\infty - \mathbf{u}_{n+1}^{k+1}|| \leq \epsilon\,||\mathbf{u}_{n+1}^k - \mathbf{d}_{n+1}|| \quad (45)$$

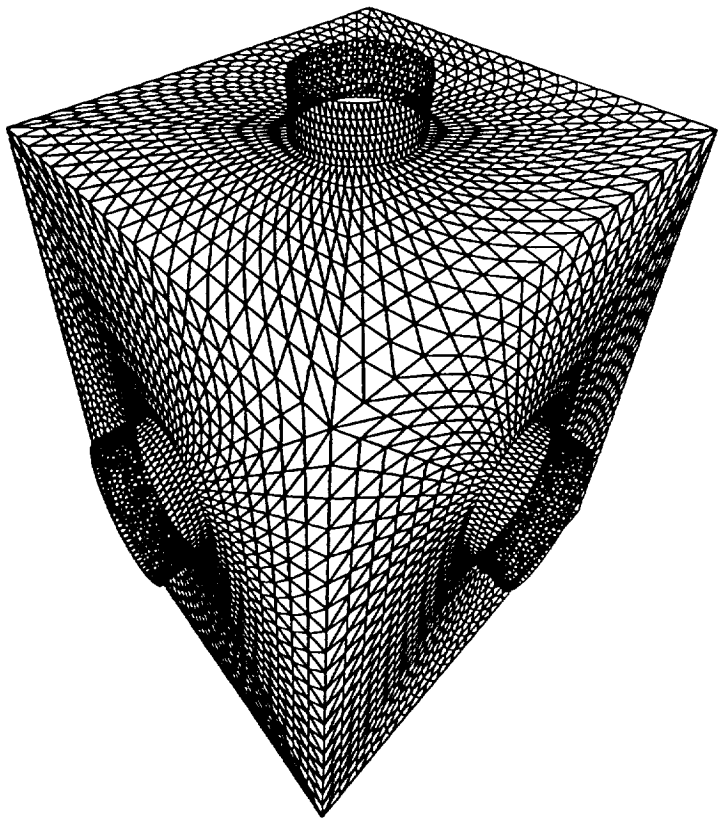

FIG. 7 *Finite element discretization of a space antenna connector (from a finer mesh)*

8.2 Reference serial performance results

First, performance results are reported for the direct method (LDL^T factorization) and the CG algorithm with diagonal scaling (Jacobi-PCG) applied to the un-decomposed problem (TABLE 5). These results will later serve as reference serial performance results. Memory consumption is measured in millions of 64 bit words (MW). MFLOPS (Million FLoating-point OPerations per Second) are reported in order to distinguish and independently assess the numerical and implementational performances. A sparse data structure is used for the Jacobi-PCG algorithm.

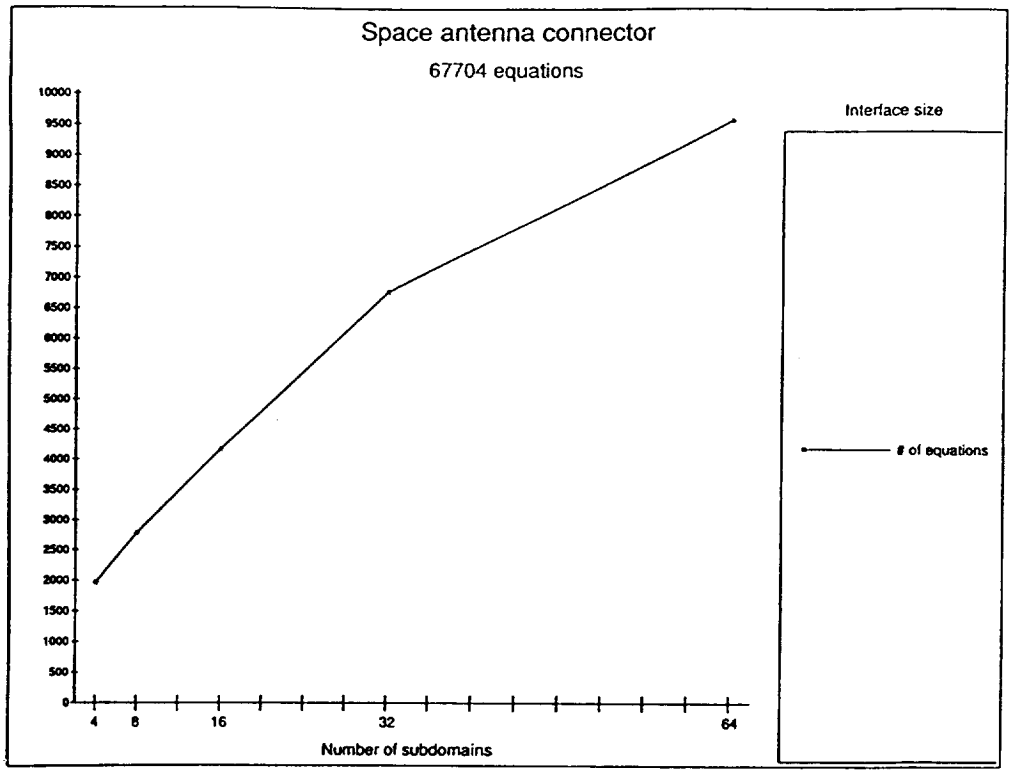

FIG. 8 *Growth of the interface problem*

TABLE 6

Space antenna connector - FETI($\widetilde{\mathbf{L}}_I^{t^{-1}}$) *serial performance results*

67704 equations - $\Delta t = \frac{T_3}{15} = 0.0123s.$

Preconditioner: $\widetilde{\mathbf{L}}_I^{t^{-1}}$

Convergence criterion: $\frac{||\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)})\,\Delta\mathbf{u}_{n+1}^{(k+1)} - \mathbf{r}(\mathbf{u}_{n+1}^{(k)})||_2}{||\mathbf{r}(\mathbf{u}_{n+1}^{(k)})||_2} \leq 10^{-4}$

CRAY Y-MP (single processor)

N_s	memory	CPU FAC	MFLOPS FAC	# of itr.	CPU /itr. PCG	CPU PCG	MFLOPS PCG	CPU TOT
4	18 MW	19.0 s.	165	159	0.43 s.	68.4 s.	130	87.4 s.
8	16 MW	13.6 s.	150	200	0.39 s.	78.0 s.	110	91.6 s.
16	12 MW	9.6 s.	130	267	0.36 s.	96.1 s.	105	105.7 s.
32	10 MW	6.7 s.	105	456	0.34 s.	155.0 s.	90	161.7 s.
64	9 MW	5.0 s.	85	717	0.33 s.	236.6 s.	75	241.6 s.

Jacobi-PCG global algorithms. In the best case (4 subdomains), the FETI method with the lumping preconditioner is three times faster than the global direct solver and four times faster than the global Jacobi-PCG algorithm.

- The primal preconditioner reduces the number of iterations performed by the lumping preconditioner by a factor of 1.4 in the [4-16] subdomains range, and by a factor of 1.2 in the [32-64] subdomains range. However, a CG iteration with the primal preconditioner is 1.8 times more expensive than a CG iteration with the lumping preconditioner, so that the lumping preconditioner is overall more efficient.
- The performance of both FETI algorithms deteriorates when the number of subdomain is increased (FIG. 9). We refer to this phenomenon as the numerical H non-scalability of the FETI methodology, Clearly, the superconvergence behavior discussed in Section 3 seems to disappear in the case of fine mesh partitions and the number of iterations seems to grow linearly with the interface size (FIG. 8). However, the reader should note that for the above structural dynamics problem, increasing the number of subdomains from 4 to 64 increases the number of iterations by a factor of 4.5, but increases the CPU time by a factor of 2.7 only (TABLE 6). This is because the cost of a PCG iteration decreases with the size of a subdomain. Moreover, since the vector speed drops from 130 MFLOPS in the 4 subdomain case to 75 MFLOPS in the 64 subdomain case, increasing the number of subdomains from 4 to 64 actually increases the operation count of the FETI method by a factor of $2.7 \times 75/130 = 1.6$ only.

a tree communication algorithm [25] is used for broadcasting at each factorization step the pivotal column to all of the processors, and a node clustered wrap mapping algorithm rather than a single degree of freedom wrap mapping algorithm is used. The distributed data structure described in [24] is augmented with a pointer array that identifies the last active column of each structural equation. On the CRAY Y-MP/8 system, the forward and backward substitutions are serialized. On the iPSC-860/128 multiprocessor, only the backward substitution is serialized. These serializations are performed because of well-known mapping, synchronization, and communication bottlenecks in the parallel solution of skyline triangular systems [23, 24].

The performances of the above parallel solvers are assessed with the nonlinear transient analysis of two structural systems: (a) the space antenna connector described in Section 8.1, and (b) a stiffened wing panel from the V22 tiltrotor aircraft (based on the panel described in Davis, Krishnamurthy, Stroud and McCleary [26]) (FIG. 10). The finite element model depicted in FIG. 10 contains 9486 nodes, 9136 4-node shell elements with 6 d.o.f. per node, and a total of 54216 d.o.f.

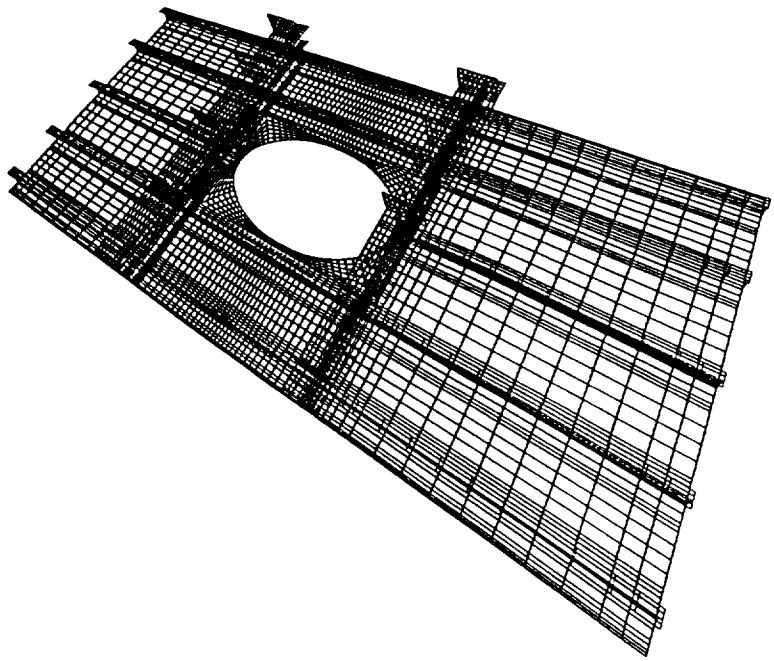

FIG. 10 *Finite element discretization of a stiffened wing panel*

TABLE 9

Space antenna connector - performance results on the CRAY Y-MP/8 system

67704 equations - $\Delta t = \frac{T_8}{15} = 0.0123s.$

FETI preconditioner: $\widetilde{\mathbf{L}}_I^{t^{-1}}$

Convergence criterion: $\frac{||\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)})\ \Delta\mathbf{u}_{n+1}^{(k+1)} - \mathbf{r}(\mathbf{u}_{n+1}^{(k)})||_2}{||\mathbf{r}(\mathbf{u}_{n+1}^{(k)})||_2} \leq 10^{-4}$

N_p	N_s	# of itr. FETI	# of itr. Jacobi-PCG	CPU FETI	CPU Jacobi-PCG	CPU direct
1	1	—	3320	—	345.6 s.	253.0 s.
4	4	159	3320	24.3 s.	93.9 s.	70.3 s.
8	8	200	3320	13.7 s.	50.2 s.	39.5 s.

TABLE 10

Stiffened wing panel - performance results on the CRAY Y-MP/8 system

54216 equations - $\Delta t = \frac{T_8}{15} = 0.00179s.$

FETI preconditioner: $\widetilde{\mathbf{L}}_I^{t^{-1}}$

Convergence criterion: $\frac{||\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)})\ \Delta\mathbf{u}_{n+1}^{(k+1)} - \mathbf{r}(\mathbf{u}_{n+1}^{(k)})||_2}{||\mathbf{r}(\mathbf{u}_{n+1}^{(k)})||_2} \leq 10^{-4}$

N_p	N_s	# of itr. FETI	# of itr. Jacobi-PCG	CPU FETI	CPU Jacobi-PCG	CPU direct
1	1	—	4775	—	418.6 s.	259.7 s.
4	4	300	4775	36.5 s.	114.0 s.	72.1 s.
8	8	396	4775	20.9 s.	60.8 s.	40.6 s.

All algorithms are reported to achieve good speed-ups. This is essentially because the CRAY Y-MP/8 system has a shared memory and a relatively small number of processors. For the antenna connector problem, the FETI method is shown to outperform the direct solver by a factor of three and the Jacobi-PCG algorithm by a factor of four. The wing panel structure apparently generates a stiffer problem: the FETI method converges with a larger number of iterations than for the connector problem, but still outperforms the direct solver by a factor of two and the Jacobi-PCG algorithm by a factor of three.

9.2. Performance results on the iPSC-860/128 multiprocessor

The performance results measured on an iPSC-860/128 Touchstone Gamma multiprocessor are summarized in TABLE 11 for the antenna connector problem and in TABLE 12 for the wing panel problem. A minimum of 32 and 64 processors are needed to accommodate the memory requirements of the FETI method and the direct solver, respectively. Our current implementation of the FETI algorithm on on this parallel processor allows only one processor per subdomain. Therefore, it is important to note that the FETI methodology is benchmarked here in the worst conditions, given that its intrinsic performance deteriorates in the presence of large numbers of subdomains (Section 8.3).

TABLE 11

Space antenna connector - performance results on the iPSC-860/128 system

67704 equations - $\Delta t = \frac{T_3}{15} = 0.0123s$.

FETI preconditioner: $\widetilde{\mathbf{L}}_I^{t^{-1}}$

Convergence criterion: $\frac{\|\widetilde{\mathbf{K}}^t(\mathbf{u}_{n+1}^{(k)})\ \Delta\mathbf{u}_{n+1}^{(k+1)} - \mathbf{r}(\mathbf{u}_{n+1}^{(k)})\|_2}{\|\mathbf{r}(\mathbf{u}_{n+1}^{(k)})\|_2} \leq 10^{-4}$

N_p	N_s	# of itr. FETI	# of itr. Jacobi-PCG	CPU FETI	CPU Jacobi-PCG	CPU direct
32	32	456	3320	144.0 s.	229.4 s.	—
64	64	717	3320	144.1 s.	144.8 s.	462.6 s.

TABLE 13

Space antenna connector - compute v.s. send v.s. receive on the iPSC-860/128 system

algorithm	N_p	compute	send	receive	dot product	total CPU time
FETI ($\widetilde{L}_I^{t^{-1}}$)	64	91.1 s.	1.0 s.	9.0 s.	43.0 s.	144.1 s.
direct (factor)	64	54.4 s.	25.4 s.	312.0 s.	—	728.1 s.
direct (forward)	64	1.4 s.	0.1 s.	14.3 s.	—	21.5 s.
direct (backward)	64	0.4 s.	1.3 s.	53.3 s.	—	193.7 s.
direct (total)	64	56.2 s.	26.8 s.	379.6 s.	—	462.6 s.

Clearly, the elapsed CPU time reported for the parallel direct solver is essentially spent in synchronization and interprocessor communication. On the other hand, the FETI method is shown to be communication efficient. The dot products performed during the solution of the interface problem are timed separately because they are implemented via calls to the iNTEL system routine **gdsum**. We could not evaluate the repartition of these timings between compute, send and receive. If we can assume that they are equally distributed between computation and communication, then we can conclude that only 21.8% of the total CPU time reported for the FETI method is spent in synchronization interprocessor communication.

The performance results summarized in TABLES 9-13 also demonstrate that the iPSC-860/128 Touchstone Gamma multiprocessor cannot compete with the CRAY Y-MP/8 system as far as direct solvers are concerned. However, for explicit-like computations such as those characterizing the FETI and Jacobi-PCG algorithms, 32 processors of the iPSC-860/128 system are shown to outperform one CRAY Y-MP processor.

10. Circumventing the numerical H non-scalability

The H non-scalability property characterizing the FETI methodology and discussed in Section 8 requires keeping the number of subdomains relatively small in order to obtain the best possible performance. However, increasing the number of subdomains while keeping the mesh size constant seems a priori attractive because: (a) it reduces the cost of the local problems, and (b) it also reduces the cost per iteration of the interface problem. Nevertheless, the results reported in TABLE 6 clearly indicate that refining beyond a certain number of subdomains the mesh partition does not significantly reduce any further the computational costs of neither the subdomain nor the interface problems. This can be explained by the fact that when the number of subdomains is increased, the subdomain

11. Closure

A memory efficient domain decomposition (DD) method for implicit transient dynamics has been presented. It is based on the concept of Finite Element Tearing and Interconnecting (FETI) developed by Farhat and Roux [8]. Two subdomain-by-subdomain preconditioners with direct mechanical interpretations have been formulated. The first preconditioner, called here a lumping preconditioner, is constructed as the sum of the projections on the subdomain interfaces of the inverses of the subdomain transient operators. The second preconditioner, called here a primal preconditioner, is constructed as the sum of the exact inverses of the subdomain transient operators. When preconditioned with the lumping operator, the interface problem behaves as if its condition number is independent of the mesh size. When preconditioned with the primal preconditioner, the interface problem has a condition number that is weakly dependent on the mesh size. Therefore, from a mathematical viewpoint, the primal preconditioner is optimal. However, the interface preconditioner is cheaper and computationally more efficient. For coarse mesh partitions, we have shown that the proposed FETI method can outperform on the CRAY Y-MP/8 multiprocessor the parallel direct solver and the parallel conjugate gradient algorithm with diagonal scaling by factors of 3 and 4, respectively. For fine mesh partitions, say more than 32 subdomains, the performance of the FETI algorithm degrades — and this is the case for most DD methods. Therefore, when working with a massively parallel processor, we recommend to keep the number of subdomains as small as possible and allocate more than one processor per subdomain. However, even in the worst case of one processor per subdomain, the FETI method still outperforms the parallel direct solver on the iPSC-860/128 system because of its low communication costs. Finally, the results we have reported for two realistic structural dynamics problems indicate that 32 processors of an iPSC-860 Gamma system can outperform a CRAY Y-MP processor.

Acknowledgments

The authors acknowledge partial support by RNR NAS at NASA Ames Research Center under Grant NAG 2-827, by the National Science Foundation under Grant ASC-8717773, and by the NASA Langley Research Center under Grant NAG-1536427.

References

[1] T.J.R. Hughes, R.M. Ferencz and J.O. Hallquist, Large-scale vectorized implicit calculations in solid mechanics on a Cray X-MP/48 utilizing EBE precon-

[14] A. van der Sluis and A. van der Vorst, The rate of convergence of conjugate gradients, Numerische Mathematik 48 (1986) 543-560.

[15] J. Mandel, *Private Communication* (1992).

[16] C. Farhat, A Lagrange multiplier based divide and conquer finite element algorithm, J. Comput. Sys. Engrg. 2 (1991) 149-156.

[17] C. Farhat, C. Felippa and M. Militello, A hybrid substructuring method and an adaptive refinement scheme for the distributed solution of three-dimensional structural problems, in: Proc. Eighth International Conference on Vehicle Structural Mechanics and Computer Aided Engineering (1992) 179-199.

[18] B. N. Parlett, The Symmetric Eigenvalue Problem, Prentice Hall, N. J. (1980).

[19] P. E. Bjordstad and O. B. Widlund, Iterative methods for solving elliptic problems on regions partitioned into substructures, SIAM J. of Num. Anal. 23 (1986) 1097-1120.

[20] G. Dahlquist and A. Bjorck, Numerical Methods, Prentice Hall, N. J. (1974)

[21] W. Chan and A. George, A linear time implementation of the Reverse Cuthill Mc Kee algorithm," BIT. 20 (1980) pp. 8-14.

[22] C. Farhat, A simple and efficient automatic FEM domain decomposer, Comput. & Struct. 28 (1988) pp. 579-602.

[23] C. Farhat, Redesigning the skyline solver for parallel/vector supercomputers, Internat. J. High Speed Comput. 2 (1988) pp. 223-238.

[24] C. Farhat and E. Wilson, A parallel active column equation solver, Comput. & Struct. 28 (1988) pp. 289-304.

[25] M. A. Baddourah *Private Communication* (1992).

[26] D. D. Davis, T. Krishnamurthy, W. J. Stroud and S. L. McCleary, An accurate nonlinear finite element analysis and test correlation of a stiffened composite wing panel, American Helicopter Society National Technical Specialists' Meeting on Rotorcraft Structures, Williamsburg, Virginia, Oct. 29-31 (1991).